Huge Data Sets and the Frontiers of Computational Feasibility

Edward J. Wegman

Technical Report No. 110
November, 1994

Center for Computational Statistics

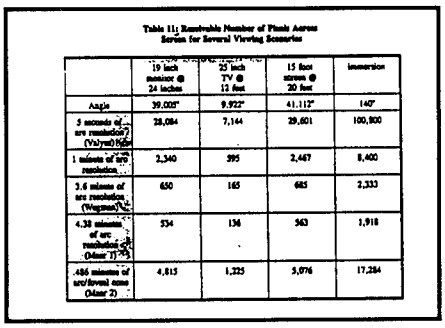

Table 11: Resolvable Number of Pixels Across Screen for Several Viewing Scenarios

	19 inch monitor @ 24 inches	35 inch TV @ 12 feet	15 foot screen @ 20 feet	immersion
Angle	39.005°	9.922°	41.112°	140°
5 seconds of arc resolution (Vaughan)	28,084	7,144	29,601	100,800
1 minute of arc resolution	2,340	595	2,467	8,400
3.6 minutes of arc resolution (Wegman)	650	165	685	2,333
4.38 minutes of arc resolution (Miner 1)	534	136	563	1,918
.486 minutes of arc/foveal zone (Miner 2)	4,815	1,225	5,076	17,284

George Mason University
Fairfax, VA 22030

CENTER FOR COMPUTATIONAL STATISTICS
RECENT TECHNICAL REPORTS

TR 94. Mark C. Sullivan and Edward J. Wegman, Correlation Estimators Based on Simple Nonlinear Transformations, February, 1994, To appear ***IEEE Transactions on Signal Processing.***

TR 95. Mark C. Sullivan and Edward J. Wegman, Normalized Correlation Estimators Based on Simple Nonlinear Transformations, March, 1994.

TR 96. Kathleen Perez-Lopez and Arun Sood, Comparison of Subband Features for Automatic Indexing of Scientific Image Databases, March, 1994.

TR 97. Wendy L. Poston and Jeffrey L. Solka, A Parallel Method to Maximize the Fisher Information Matrix, June, 1994.

TR 98. Edward J. Wegman and Charles A. Jones, Simulating a Multi-target Acoustic Array on the Intel Paragon, June, 1994.

TR 99. Barnabas Takacs, Edward J. Wegman and Harry Wechsler, Parallel Simulation of an Active Vision Model, June, 1994.

TR 100. Edward J. Wegman and Qiang Luo, Visualizing Densities, October, 1994.

TR 101. Daniel B. Carr, Converting Tables to Plots, October, 1994.

TR 102. Julia Corbin Fauntleroy and Edward J. Wegman, Parallelizing Locally-Weighted Regression, October, 1994.

TR 103. Daniel B. Carr, Color Perception, the Importance of Gray and Residuals on a Choropleth Map, October, 1994.

TR 104. David J. Marchette, Carey E. Priebe, George W. Rogers and Jeffrey L. Solka, Filtered Kernel Density Estimation, October, 1994.

TR 105. Jeffrey L. Solka, Edward J. Wegman, Carey E. Priebe, Wendy L. Poston and George W. Rogers, A Method to Determine the Structure of an Unknown Mixture Using the Akaike Information Criterion and the Bootstrap, October, 1994.

TR 106. Wendy L. Poston, Edward J. Wegman, Carey E. Priebe and Jeffrey L. Solka, A Contribution to the Theory of Robust Estimation of Multivariate Location and Shape: EID, October, 1994.

TR 107. Clifton D. Sutton, Tree Structured Density Estimation, October, 1994.

TR 108. Charles A. Jones, Simulating a Multi-target Acoustic Array on the Intel Paragon (M.S. Thesis), October, 1994.

TR 109. Leonard B. Hearne and Edward J. Wegman, Fast Multidimensional Density Estimation based on Random-width Bins, October, 1994.

TR 110. Edward J. Wegman, Huge Data Sets and the Frontiers of Computational Feasibility, November, 1994.

TR 111. Winston C. Chow, Fractional Process Modeling, November, 1994.

Huge Data Sets and the Frontiers of Computational Feasibility[1]

Edward J. Wegman

Center for Computational Statistics
George Mason University
Fairfax, VA 22030

Abstract

Recently, Huber (1994) offered a taxonomy of data set sizes ranging from tiny (10^2 bytes) to huge (10^{10} bytes). This taxonomy is particularly appealing because it quantifies the meaning of tiny, small, medium, large and huge. Indeed, some investigators consider 300 small and 10,000 large while others consider 10,000 small. In Huber's taxonomy, most statistical and visualization techniques are computationally feasible with tiny data sets. However with larger data sets computers run out of computational horsepower and graphics displays run out of resolution fairly quickly. In this paper, we discuss aspects of data set size and computational feasibility for general classes of algorithms in the context of CPU performance, memory size, hard disk capacity, screen resolution and massively parallel architectures. We discuss some strategies such as recursive formulations which mitigate the impact of size. We also discuss the potential for scalable parallelization which will mitigate the effects of computational complexity.

[1]This research was supported by the Army Research Office under contract number DAAH04-94-G-0267 and by the Office of Naval Research under contract number N00014-92-J-1303.

1. Introduction

In Minamisaku, Japan, an array of eighty-four antennas monitors the surface of the sun for evidence of solar flares (Nakajima et al., 1994). The task of converting microwave signal data into images of the solar surface requires an enormous number of computations. The imaging routines involve the estimation of the correlation matrix which requires $84 \times (84-1)/2 = 3486$ multiply and accumulate operations for every data sample collected. The data is downconverted from 17 GHz to a complex baseband format with a sample rate of 40 Mhz. Processing is by necessity real-time so that the number of real-time multiplications per second required is $3486 \times 4 \times 40 \times 10^6 = 557.76$ billion. Massive data sets with even relatively simple operations clearly test the limit of even the most ambitious general purpose computer hardware. The $\frac{1}{2}$ teraflop performance that the antenna array processing application demands is greater than any current existing supercomputer can produce and is certainly involves data analysis well within the parameters of the so-called grand challenge, high performance computing initiative. In the case of the Minamisaku antenna array, the solution is to create a parallel array of special purpose processing chips that compute a hard-limited version of the usual product-moment correlation estimator. Hard limiting simply means that the actual data value is replaced by $+1$ or -1 depending on the sign of the actual data value. The correlation estimator then can be computed by simply bit flipping and accumulating. Such a solution involves a significant tradeoff between desirable statistical properties and computational feasibility.

It is probably a truism to say that most statisticians have not worried very much about the computational feasibility of the algorithms they have invented. The possible exception to this statement is in connection with nonparametric density estimation. We believe that the lack of concern about computational feasibility generally flows from the mind set on sample sizes that tends to regard a data set with 10,000 observations as a very big data set. For some time now we have argued, (see Wegman 1988a,b), that electronic instrumentation and modern computing resources have significantly altered the size and character of some classes of data sets in that they tend to be much larger, much higher dimensional and much less homogeneous than traditional statistical data sets. More recently, Huber (1992, 1994) has proposed a taxonomy of *large* data sets which he characterizes as given below in Table 1.

Descriptor	Data Set Size in Bytes	Storage Mode
Tiny	10^2	Piece of Paper
Small	10^4	A Few Pieces of Paper
Medium	10^6	A Floppy Disk
Large	10^8	Hard Disk
Huge	10^{10}	Multiple Hard Disks e.g. RAID Storage
Ridiculous	10^{12}	Robotic Magnetic Tape Storage Silos

Table 1. The Huber Taxonomy of Data Set Sizes

Responsibility for the last descriptor must not be laid upon Professor Huber, but upon us. However, this last descriptor gives the order of magnitude (i.e. a terabyte of data) that is routinely discussed in connection with the so-called Grand Challenge Problems, the High Performance Computing and Communication Initiative (HPCC) and the National Information Infrastructure (NII). It is also the magnitude of data that is expected daily with the Earth Observing System (EOS) now underway under the aegis of NASA and, clearly, it would not take very long with the Minamisaku antenna array to accumulate data whose sample size is of this order of magnitude. The point is that it is now appropriate to consider the computational impact of large and huge and perhaps even terabyte data sets. Given this taxonomy of data set sizes, supposed computational complexity of algorithms [e.g. $O(n^{1/2})$, $O(n)$, $O(n\ log(n))$, $O(n^{3/2})$ and $O(n^2)$], and, finally, some CPU speeds, it is possible to portray the limits of interactive feasibility and the limits of computational feasibility.

In this paper, I intend to explore some limits of both computational feasibility and visualization feasibility. The intent is to define in some sense the parameters for which computational feasibility and visualization feasibility are at their limits. These parameters thus define the settings in which it is profitable to expend intellectual effort.

2. Computational Complexity and Order of Magnitude Considerations

Huber's taxonomy as outlined above focuses on the number of bytes in a data set. For our current discussion, we are mainly interested in order-of-magnitude

considerations. In fact, a single one-dimensional observation is two to eight bytes depending on the data structure used for the underlying data set. A 32-bit word will occupy four bytes while a 64-bit word will occupy eight bytes. This scalar multiple will obviously impact the timing of a computation, but will not generally impact the order of magnitude seriously. So for our order of magnitude considerations, we make the simplifying assumption that an observation is to be taken as one byte.

Often data is arranged in a matrix form with r rows and c columns where $n = r \times c$. For higher dimensional data, say d-dimensional, $d = c$. Often, we will take $r \propto \sqrt{n}$. Our general strategy is to make some complexity assumptions. This gives us the number of computations required as a function of the number of observations. Using the Huber taxonomy, we can, then, at least to an order of magnitude except for a scalar constant, calculate the total number of operations required for a given procedure on a data set. The total number of operations (except for the scalar constant) is outlined in Table 2. It is worthwhile to point out some operations a statistician would normally perform together with their complexity. This is given below in Table 3.

Having said that, I will ignore the impact of the scalar multiplier, it is worth exploring the range that this scalar multiplier can take on. The calculation of a simple mean and of a kernel density estimator are both $O(n)$ complexity algorithms. Let us consider how they differ in terms of their scalar constants. The formula

$$\overline{X} = \sum_{i=1}^{n} X_i$$

has $(n-1)$ adds and 1 multiply for a total complexity of exactly n. Thus the scalar constant is just 1. By contrast the calculation for the kernel density estimator is given by

$$\hat{f}(x) = \frac{1}{nh_n} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h_n}\right) \tag{2.1}$$

has considerably more complexity. The expression $\left(\frac{x - X_i}{h_n}\right)$ has one subtract and one divide for a complexity of 2. However, the kernel $K(\cdot)$ is often a transcendental function. For example if $K(\cdot)$ is a Gaussian kernel, it is often approximated with a rational 10^{th} degree polynomial. (See Abramowitz and Stegun, 1965). The usual 10^{th} polynomial will have 5 powers, 5 multiplies, 5 adds and one divide for a total complexity of 16. Now $\sum_{i=1}^{n}(\cdot)$ as before involves $(n-1)$ adds. Finally, there is one additional multiply yielding a total complexity of $18(n-1)+1$ for each value of x. The density

Table 2: Number of Operations for Algorithms of Various Computational Complexities and Various Data Set Sizes

n	$n^{1/2}$	n	$n \log(n)$	$n^{3/2}$	n^2
tiny	10	10^2	$2x10^2$	10^3	10^4
small	10^2	10^4	$4x10^4$	10^6	10^8
medium	10^3	10^6	$6x10^6$	10^9	10^{12}
large	10^4	10^8	$8x10^8$	10^{12}	10^{16}
huge	10^5	10^{10}	10^{11}	10^{15}	10^{20}

estimator is often computed on a grid of m elements on a side. If d is the dimension, then there are m^d x's to consider yielding a total complexity of $m^d[18(n-1)+1] = O(n)$. In this case the scalar multiplier is $18m^d$. (Strictly speaking (2.1) is the univariate formula. However, the reader can verify that the complexity computation is still essentially correct if we substitute the multivariate formula.) for modest cases, say $m = 10$ and $d = 5$, the scalar multiplier is $18 \times 10^5 = 1.8 \times 10^6$. Thus even though the scalar multiplier is not a function of n it may still dramatically impact the computations so that in reading Tables 4 through 9, the reader should bear in mind that the scalar multiplier may change an algorithm from being "feasible" to "infeasible" for a particular configuration.

$O(r)$, $O(n^{1/2})$	Plot a scatterplot
$O(n)$	Calculate means, variances, kernel density estimates
$O(n\ log(n))$	Calculate fast Fourier transforms
$O(nc)$	Calculate singular value decomposition of an $r \times c$ matrix; solve a multiple linear regression
$O(nr)$, $O(n^{3/2})$	Solve a clustering algorithm with $r \propto \sqrt{n}$
$O(n^2)$	Solve a clustering algorithm with c fixed and small so that $r \propto n$.

Table 3. Algorithmic Complexity

With these basic considerations in hand, I am now able to use data set size, computational complexity, and machine speeds to compute times to completion.

3. Interactive and Computational Feasibility

Table 2 essentially gives me the number of floating point operations (within a scalar multiple) needed to complete a given complexity algorithm. By assuming certain realistic computer configurations and then dividing the elements of Table 2 by the megaflop performance of those computer configurations, I am able to calculate times required. I will assume four basic configurations: 1) a PC based on Intel's Pentium processor, 2) a Silicon Graphics Onyx Workstation, 3) an Intel Paragon XP/S A4, and

Table 4: Computational Feasibility on a Pentium PC
10 megaflop performance assumed

n	$n^{1/2}$	*n*	*n log(n)*	$n^{3/2}$	n^2
tiny	10^{-6} *seconds*	10^{-5} *seconds*	$2x10^{-5}$ *seconds*	*.0001 seconds*	*.001 seconds*
small	10^{-5} *seconds*	*.001 seconds*	*.004 seconds*	*.1 seconds*	*10 seconds*
medium	*.0001 seconds*	*.1 seconds*	*.6 seconds*	*1.67 minutes*	*1.16 days*
large	*.001 seconds*	*10 seconds*	*1.3 minutes*	*1.16 days*	*31.7 years*
huge	*.01 seconds*	*16.7 minutes*	*2.78 hours*	*3.17 years*	*317,000 years*

Table 5: Computational Feasibility on a Silicon Graphics Onyx Workstation
300 megaflop performance assumed

n	$n^{1/2}$	*n*	*n log(n)*	$n^{3/2}$	n^2
tiny	$3.3x10^{-8}$ *seconds*	$3.3x10^{-7}$ *seconds*	$6.7x10^{-7}$ *seconds*	$3.3x10^{-6}$ *seconds*	$3.3x10^{-5}$ *seconds*
small	$3.3x10^{-7}$ *seconds*	$3.3x10^{-5}$ *seconds*	$1.3x10^{-4}$ *seconds*	$3.3x10^{-3}$ *seconds*	*.33 seconds*
medium	$3.3x10^{-6}$ *seconds*	$3.3x10^{-3}$ *seconds*	*.02 seconds*	*3.3 seconds*	*55 minutes*
large	$3.3x10^{-5}$ *seconds*	*.33 seconds*	*2.7 seconds*	*55 minutes*	*1.04 years*
huge	$3.3x10^{-4}$ *seconds*	*33 seconds*	*5.5 minutes*	*38.2 days*	*10,464 years*

Table 6: Computational Feasibility on an Intel Paragon XP/S A4
4.2 gigaflop performance assumed

n	$n^{1/2}$	n	$n \log(n)$	$n^{3/2}$	n^2
tiny	2.4×10^{-9} seconds	2.4×10^{-8} seconds	4.8×10^{-8} seconds	2.4×10^{-7} seconds	2.4×10^{-6} seconds
small	2.4×10^{-8} seconds	2.4×10^{-6} seconds	9.5×10^{-6} seconds	2.4×10^{-4} seconds	.024 seconds
medium	2.4×10^{-7} seconds	2.4×10^{-4} seconds	.0014 seconds	.24 seconds	4.0 minutes
large	2.4×10^{-6} seconds	.024 seconds	.19 seconds	4.0 minutes	27.8 days
huge	2.4×10^{-5} seconds	2.4 seconds	24 seconds	66.7 hours	761 years

Table 7: Computational Feasibility on a Teraflop Grand Challenge Computer
1000 gigaflop performance assumed

n	$n^{1/2}$	n	$n \log(n)$	$n^{3/2}$	n^2
tiny	10^{-11} seconds	10^{-10} seconds	2×10^{-10} seconds	10^{-9} seconds	10^{-8} seconds
small	10^{-10} seconds	10^{-8} seconds	4×10^{-8} seconds	10^{-6} seconds	10^{-4} seconds
medium	10^{-9} seconds	10^{-6} seconds	6×10^{-6} seconds	.001 seconds	1 second
large	10^{-8} seconds	10^{-4} seconds	8×10^{-4} seconds	1 second	2.8 hours
huge	10^{-7} seconds	.01 seconds	.1 seconds	16.7 minutes	3.2 years

Table 8: Types of Computers for Interactive Feasibility
Response Time < 1 second

n	$n^{1/2}$	n	$n \log(n)$	$n^{3/2}$	n^2
tiny	*PC*	*PC*	*PC*	*PC*	*PC*
small	*PC*	*PC*	*PC*	*PC*	*SC*
medium	*PC*	*PC*	*PC*	*SC*	*TC*
large	*PC*	*WS*	*SC*	*TC*	---
huge	*PC*	*SC*	*TC*	---	---

Table 9: Types of Computers for Feasibility
Response Time < 1 week

n	$n^{1/2}$	n	$n \log(n)$	$n^{3/2}$	n^2
tiny	*PC*	*PC*	*PC*	*PC*	*PC*
small	*PC*	*PC*	*PC*	*PC*	*PC*
medium	*PC*	*PC*	*PC*	*PC*	*PC*
large	*PC*	*PC*	*PC*	*PC*	*TC*
huge	*PC*	*PC*	*PC*	*SC*	---

PC=Personal Computer; WS=Workstation; SC=Supercomputer; TC=Teraflop Computer

the yet-to-be-produced Teraflop Supercomputer. The 90 Mhz Pentium processor is capable of approximately 10 megaflops. The times outlined in Table 4 for this Pentium PC are compute times for the *CPU* only and do not reflect i/o times or any other overhead times. The machines are often capable of holding 256 MB of RAM.

The Silicon Graphics Onyx is a new generation of machine. The CPU is an R4400 chip double-clocked to 200 megahertz. The machine has the capability of holding 2 gigabytes of RAM and has a peak speed rating of 300 megaflops. Peak speeds are defined in terms of the number of floating point operations a modern pipelined chip can carry out simultaneously. Since most algorithms do not even come close to having the required mix of operations to achieve peak speed, this rating is, in effect, the upper speed limit beyond which the computer manufacturer guarantees you will not be able to go. My Table 5 assumes that we could actually achieve this peak speed.

The third machine considered is an Intel Paragon XP/S A4. The machine is the current generation of Intel's Supercomputer series. The A4 model is a 61-node academic machine with a peak speed rating of 4.2 gigaflops. The machine is based on the Intel 860 processor which has the capability of performing one floating point add every clock cycle but requiring two clock cycles for a floating multiply. The machine has a distributed memory of just under 1 gigabyte of RAM. Because it is a distributed memory machine, there is additional communication overhead. Although Table 6 is based on the 4.2 gigaflop peak performance, the machine is unlikely to achieve this level of performance. As of this writing, a 1,840 node Intel Paragon owned by Sandia National Labs achieved the world speed record of 143.4 double-precision gigaflops.

The teraflop computer is the stated goal of the US federal government's High Performance Computing and Communication (HPCC) initiative. The initiative was originally targeted at some grand challenges of science. Table 7 represents the compute times associated with such a teraflop computer. Currently, no such computer exists. It is worth commenting that although such a computer will probably be achieved, there are foreseeable limits on the ultimate speeds achievable. The early supercomputers (now called parallel-vector computers) were uniprocessor machines with very sophisticated processors. Speed-of-light considerations caused the manufacturers to try to make smaller and smaller processors. The increased density of the chips cause heat dissipation problems which in turn require exotic cooling methods. Ultimately the limiting size of the features on a chip will be determined by quantum effects when the number of electrons is sufficiently small. An alternate strategy has been to use

relatively inexpensive off-the-shelf microprocessors in a configuration that is now called massively parallel. Clearly as microprocessors evolve, they will begin to experience the same limitations that parallel-vector processors experienced. The only solution (at least in current electronic technology) is to have a massively parallel array of supercomputer processors. Scaling up the number of processors in the machine will also be limited by speed of light considerations. Ultimately, the quest for speed must cause us to abandon current electronic technology, and, perhaps, turn to optics.

In Tables 8 and 9, I define *interactive feasibility* to mean compute times for a given configuration of 1 second or less. This is perhaps somewhat arbitrary, but a useful limit given my order-of-magnitude considerations. Similarly, I define *computational feasibility* to mean compute times of 1 week or less. Table 8 is assembled by looking for the minimal computer configuration that will achieve the required interactive feasibility. Table 9 is assembled by looking for the minimal computer configuration which will achieve the required computational feasibility. Because of the fact that I ignored the scalar multiplier and that I used peak speed, all of these tables are extremely optimistic. This means configurations that are infeasible are really, truly infeasible. Interestingly enough, these tables also demonstrate that some relatively innocuous statistical and data analysis problems easily fit into the framework of Grand Challenge Problems in Science even though most statisticians would not consider their work to be that demanding. *What is clear is that data sets of large and huge size require intellectual attention. If statisticians don't pay attention to these problems, other scientists will.*

Some conclusions on CPU speed are also of interest. In terms of interactive feasibility, we must be drawn to the conclusion that PC's are already remarkably capable. While supercomputers and teraflop computers fill some of the niches, PC's fill the majority of the configurations. It is probably safe to conclude that existing workstations and supercomputers extend range of computing to account for scalar constants not really dealt with in Table 8. One is also tempted to conclude that teraflop computers will have comparatively little impact, especially in view of their considerable expense. Indeed, an improvement from an $O(n^2)$ to an $O(n^{3/2})$ algorithm seems to have far more dramatic impact. Also, as we shall see in the next section, even though some categories have interactive feasibility in terms of *CPU* speed, data transfer rates will limit interactive feasibility.

In terms of computational feasibility, we must again be drawn to the conclusion that PC's already remarkably capable. As before, supercomputers and teraflop computers will mainly impact scalar multipliers, but not orders of magnitude. What is particularly apparent here is the very high payoff in algorithm improvement. Indeed, here moving from a $O(n^2)$ algorithm to a $O(n^{3/2})$ algorithm can move us from a teraflop computer requirement to a PC requirement in the case of large (10^8) data sets or from computationally infeasible to an ordinary supercomputer in the case of huge (10^{10}) data sets.

4. Data Transfer Rates

While the CPU speed analysis is useful, it seems clear that i/o represents a potential bottleneck, particularly for large and huge data sets. To consider the effect of data transfer rates, we consider four technologies: 1) standard ethernet, 2) fast ethernet, 3) disk transfer and 4) cache memory transfer. Standard ethernet is a commonly available technology. The peak transfer rate for standard ethernet is 10 megabits per second. However, as in the case of peak CPU speeds, this is the maximum rate beyond which one is guaranteed not to go. TCP/IP used with ethernet is an asynchronous protocol so that when collisions occur, each machine backs off a random amount of time and resends its message. Thus in a heavily congested environment, the communication times can be considerably slower. (Anecdotally, the message-passing interconnect on the original Intel iPSC hypercube was an ethernet protocol. The typical programming paradigm was an SIMD so that most nodes tended to finish at approximately the same time. This created heavy congestion and made the Intel iPSC/1 a rather slow machine. This same problem afflicts the current PVM programming model).

Fast ethernet is representative of a host of emerging technologies including FDDI, ATM/SONET, and HiPPI to mention a few. Fast ethernet has a peak data transfer rate of 100 megabits per second. Presumably, most data sets will be stored on disk in some format. Transfer rates for hard disks vary considerably. However, the figure given in our Table 10 is a measured transfer rate for a Seagate SCSI hard drive installed in a PC. Finally, with more modern chips and large RAM memory, much or all of the data can be stored in RAM. If this is the case, the limiting factors, particularly for interactive feasibility become the cache transfer rates. The comparison in Table 10 is based on the assumption that the mother board operates with a 200 megahertz clock and that one byte can be transferred each clock cycle. Actually, the more modern 64-bit wide bus

Table 10: Transfer Rates for a Variety of Data Transfer Regimes

n	*standard ethernet 10 mega-bits/sec*	*fast ethernet 100 mega-bits/sec*	*hard disk transfer 2027 kilo-bytes/sec*	*cache transfer @ 200 megahertz*
	1.25×10^6 *bytes/sec*	1.25×10^7 *bytes/sec*	2.027×10^6 *bytes/sec*	2×10^8 *bytes/sec*
tiny	8×10^{-5} *seconds*	8×10^{-6} *seconds*	4.9×10^{-5} *seconds*	5×10^{-6} *seconds*
small	8×10^{-3} *seconds*	8×10^{-4} *seconds*	4.9×10^{-3} *seconds*	5×10^{-5} *seconds*
medium	.8 *seconds*	.08 *seconds*	.49 *seconds*	5×10^{-3} *seconds*
large	1.3 *minutes*	8 *seconds*	49 *seconds*	.5 *seconds*
huge	2.2 *hours*	13.3 *minutes*	1.36 *hours*	50 *seconds*

structures may allow for up to eight bytes per clock cycle. However, because of my earlier simplifying assumption of one byte per data item, I do not pursue this subtlety.

It is worth noting some simple observations. As noted earlier for 32-bit words, it would be necessary to multiply times in Table 10 by 4 while for 64-bit words, it would be necessary t multiply the times by 8. As noted earlier, many simple PC machines are capable of using up to 256 megabytes RAM while more sophisticated workstations and supercomputers are capable of holding 2 gigabytes or more of RAM. The range from 2.56×10^8 to 2×10^9 bytes implies that it is feasible to hold "large" data sets in RAM. However, this by itself is not sufficient since between four to ten times the data set size may required for scratch space and partial computation depending on the nature of the algorithm. In addition, many contemporary computers use 32-bit addressing. This implies that there are $2^{32} \doteq 4.294 \times 10^9$ address locations. Thus with 32-bit addressing, it is not feasible to hold "huge" data sets in RAM.

Table 10 allows us to draw the conclusion that interactive computing is feasible with "medium" (10^6) data sets and perhaps "large" (10^8) data if the data can be stored in RAM. However, as with the *CPU* feasibility tables, large and huge data sets push the limit of the technology. It is interesting to note that even with the slower technologies, the communication speeds do not limit computational feasibility. However, even when the algorithm is sufficiently simple, i.e. $O(n^{1/2})$ or $O(n)$ to suggest that the *CPU* is capable of interactive computing for large or huge data sets, the data transfer rates will prohibit interactive computing.

5. Limits of Visualization

Not so long ago, I heard a very serious proposal to develop technologically very complex visualization system which would involve multiple projection display systems with a total capacity of some 8000 by 6000 pixels in an area approximately 20 feet by 15 feet. The reason advanced for creating this system was that the fine structure in fluid dynamics computations needed such resolutions. What seemed to be lacking to me was any indication that at reasonable viewing distances, the human eye itself would be able to resolve such high resolution images. The following discussion is intended to explore some limits of visual perception and what size data sets might we hope to view directly.

The classical scatter plot is a graphical device which codes each data item as a pixel. As such, it represents something of a minimal coding and, perhaps, a useful

benchmark against which to measure our ability to directly visualize data sets with a graphic encoding. The question then becomes what is the minimal pixel size that can be resolved. One can imagine two rather distinct answers to this question. We could imagine, for example, a self-luminescent pixel shrinking so small that its image passing through the lens of our eye would have to fall between two foveal cones and thus not excite either one of them. No matter how bright the pixel was, one would not see it. Of course, defects in the focus of the lens and visual saccades are likely to spread the image so that periodically it would stimulate a cone and, perhaps therefore, seem to twinkle. More pertinent to our line of enquiry is how close can two pixels be and still be distinguished one from the other. Presumably the pixels would have to be somewhat larger than the just-visible case mentioned above.

In order to answer the second question, I undertook a relatively simple experiment. I drew three parallel black lines on a piece of white paper separated respectively by 1/2 inch and 3/4 inch. This was backlit with sunlight. I and two of my colleagues then proceeded a distance of more than 100 feet away and walked toward the subject paper until we could distinguish the three lines drawn on the paper. I knew of course what was there, but neither of my colleagues had prior information about the image on the paper. I am corrected to 20/15. All three of us stopped within a few feet of each other and our mean angular resolution was 3.6 minutes of arc. While I have not done an extensive literature search on the topic, the literature seems inconclusive. Valyus (1962) claims a figure of 5 seconds of arc while Maar (1982) suggests 4.38 minutes of arc. Maar also suggests that a foveal cone subtends 0.486 minutes of arc and that approximately nine foveal cones are required to distinguish two adjacent pixels. Based on this size, it would appear that two adjacent foveal cones would occupy approximately 1 minute of arc. It seems like the very most we could hope for is that two adjacent pixels could be mapped into two adjacent foveal cones and that the brain was sufficiently delicately wired so that the signals from those two cones could be distinguished. This seems very optimistic. Since the Maar figure of 4.38 minutes of arc is reasonably close to our own experimental result of 3.6 minutes of arc and given the size of the foveal cone, I am inclined to believe that the Valyus figure refers more precisely to the first kind of resolution, i. e. the just-visible pixel.

Having determined the angular resolution, we can then compare that result to a number of typical viewing scenarios. For example, viewing a 19-inch monitor (17-inch image) at a distance of 24 inches might be a reasonable typical computer monitor

Table 11: Resolvable Number of Pixels Across Screen for Several Viewing Scenarios

	19 inch monitor @ 24 inches	25 inch TV @ 12 feet	15 foot screen @ 20 feet	immersion
Angle	39.005°	9.922°	41.112°	140°
5 seconds of arc resolution (Valyus)	28,084	7,144	29,601	100,800
1 minute of arc resolution	2,340	595	2,467	8,400
3.6 minute of arc resolution (Wegman)	650	165	685	2,333
4.38 minutes of arc resolution (Maar 1)	534	136	563	1,918
.486 minutes of arc/foveal cone (Maar 2)	4,815	1,225	5,076	17,284

setting. It is a simple matter to compute that the viewing angle would be approximately 39.005°. Knowing the viewing angle and the angular resolution, we can calculate approximately how many pixels across the screen we could resolve. Other viewing settings include viewing a 25-inch television at 12 feet, viewing a 15-foot diagonal screen at 20 feet, and what we call *immersion.* We have in mind a virtual reality-type setting in which the image completely surrounds the viewer so that the viewing angle is limited to the limits of peripheral vision. We take this to be about 140°. Of course, much of the scene being viewed would actually be imaged on lower resolution rods rather than foveal cones, so that the extrapolation based on the resolution of foveal cones would be optimistic.

Table 11 contains estimates of the number of resolvable pixels for four viewing scenarios; 1) 19 inch monitor at 24 inches, 2) 25 inch television at 12 feet, 3) 15 foot screen at 20 feet and 4) immersion; and for five assumed resolutions; 1) Valyus's 5 seconds of arc, 2) 1 minute of arc, 3) Wegman's 3.6 minutes of arc, 4) Maar's 4.38 minutes of arc, and 5) Maar's 0.486 minutes of arc per foveal cone. Based on Table 11, I do some scenarios we might consider.

Scenarios

- **Typical high resolution workstations,**
  $1280 \times 1024 = 1.31 \times 10^6$ **pixels**
- **Realistic using Wegman, immersion, 4:5 aspect ratio,**
  $2333 \times 1866 = 4.35 \times 10^6$ **pixels**
- **Very optimistic using $1'$ arc, immersion, 4:5 aspect ratio,**
  $8400 \times 6720 = 5.65 \times 10^7$ **pixels**
- **Wildly optimistic using Maar(2), immersion, 4:5 aspect ratio,**
  $17{,}284 \times 13{,}828 = 2.39 \times 10^8$ **pixels**

Some conclusions follow immediately. Using single pixel coding, it seems unlikely that under any circumstances one would be able to visualize large (10^8) or huge (10^{10}) data sets. Medium (10^6) data sets may just be feasible to visualize, but even this is not particularly clear cut since heavy overplotting is likely to complicate direct viewing. What is very clear is that traditional graphical exploratory data analysis tools do not scale up to large and huge data sets. What is also clear is than some of the obvious ways of paring these data sets down, e.g. clustering, discriminant analysis,

principal components, are computationally sufficiently complex ($O(n^{3/2})$ or $O(n^2)$) that they do not offer viable alternatives. Visualization of data sets say of size 10^6 or more is a clearly a wide open field.

6. Some Concluding Suggestions and Observations

It is probably axiomatic that as the size of the data set increases, so does its complexity. In a sense it becomes lumpier and richer in structure. Very large data sets that are simple in structure are very boring and also unlikely to be collected. Applying traditional statistical methods to what in Huber's taxonomy are medium, large or huge data sets is doomed to failure. Homogeneity is almost surely gone. High-dimensional data sets probably admit topologically complicated substructures that most of us have not even imagined. Any parametric model will almost surely be rejected by any hypothesis testing procedure. Fashionable techniques such as bootstrapping are computational too complex to be seriously considered for many of these data sets. Random subsampling and dimensional reduction techniques are very likely to hide the very substructures that may be pertinent to the correct analysis of the data. I hope it is not too outrageous to draw the conclusion that nonparametric analytic techniques and graphical exploratory analysis tools are crucial to our ability to deal with such massive data sets, and, yet, as we have seen in the foregoing discussion, these techniques are the very one's which test the limits of our computing capabilities most severely. A serious intellectual effort *by statisticians* must be made to address those tools and data set sizes which are at the limit of present and future computational capabilities. I believe it is naive to believe that computers will always continue to improve so that we never have to seriously deal with issues of computational complexity. I believe it is also necessary for statisticians to pay attention to database structures and database techniques. If statisticians continue to regard data structures as a topic with minor relevance, then we will not be positioned to invent the new data analysis techniques and statistical methods required for 10^6 and larger data sets. I have one other policy oriented suggestion. Just as statisticians now routinely document the small sample and asymptotic properties of a new technique, e.g. optimality, bias, consistency, distribution or asymptotic distribution, we should also document, and expect to see documented, the complexity properties of the algorithm associated with the new technique.

I believe there are some useful technical suggestions to be made as well.

Visualization

SCINTILLATION In our graphical work, we have found it convenient to color code distinct observations with distinct colors. Some may object to this strategy because different colors do not have equal brightnesses so that some observations may be overlooked or appear less prominent than others. However, the point is not to look for individual observations as much as the overall structure. Having color coded observations with multiple colors, one has an opportunity to deal with overplotting by using a time multiplexing technique. If a pixel represents multiple points, stepping through the colors associated with points represented by that pixel sequentially, causes the pixel to flash. The more overplotting there is, the more rapid the flashing of the pixel. This technique focuses visual attention where the action is, so to speak, and has the potential for increasing the usable sample size by an order of magnitude or more.

DENSITY RENDERING (WITH OUTLIERS) It seems clear that for sufficiently large data sets, no single pixel coding is likely to be successful. Kernel smoothers have a computational complexity of $O(n)$ and new adaptive and binning algorithms are likely to be even more computationally efficient. Thus it is computationally feasible and even interactively feasible to use density representations in place of single pixel coding. Moreover, even with tools like scintillation mentioned above, overplotting often obscures structure which may be much better revealed by nested density contours. Four-dimensional holes in the data are an example of a structure in the data that may easily be revealed with density contours that are difficult to see with scatterplots. Wegman (1994) discusses density visualization with transparency and rendering which allows for the visualization of very complex and subtle structure. Of course, once a density estimate is computed, which may be a non-interactive computation, the density may then be interactively explored so that density rendering also expands the possibility of interactive exploration for data sets which may be too large for more traditional exploration. One obvious suggestion is to plot the density (density contours) where the data is most dense and to simply plot the outlier points where the data is less dense. The caveat is of course that our notion of outlier may have to change. After all the 1% outliers of a 10^{10} data set is itself a 10^8 data set which according to our previous discussion is not visualizable. Similarly, the .01% outliers of a 10^{10} data set is still a 10^6 data set which may still suffer badly from overplotting.

IMMERSION Virtual reality has been hyped a tremendous amount. However, as we have seen above, immersive technology has the possibility of substantially expanding the number of resolvable pixels in addition to giving the data analyst freedom to move in the third dimension. After all, the screen is two dimensional and the two-dimensional pixel count we gave above as $2333 \times 1866 = 4.35 \times 10^6$ pixels could be augmented by the third dimension by another 2333, i.e. $2333 \times 2333 \times 1866 = 1.015 \times 10^{10}$ voxels using three dimensional virtual reality techniques. They could conceivably aid us in visualizing huge data sets.

Computation

RECURSION We advocate more serious addressal of recursive algorithms especially for very large data sets. Recursion has two beneficial effects. First, not all of the data must remain in the computer memory simultaneously. Thus RAM is conserved and real-time data acquisition and processing can be addressed. For some classes of data such as our opening example of the 84 microwave antennas, raw data cannot even be stored. Thus recursive algorithms make real-time computations feasible. Moreover, even if data can be stored, the second benefit is that new data does not necessitate recomputation from scratch. For algorithms that are reasonably computationally complex, say even $O(n)$ or more, a reformulation in recursive form may make updating feasible. Moreover, a recursive formulation may allow for decisions earlier as the computation evolves.

ADAPTATION By an adaptive algorithm, I mean an algorithm which takes into account the structure of the data and when the data is simple, the algorithm may be substantially simpler. I have in mind examples like Priebe's adaptive mixtures nonparametric density estimator (Priebe, 1994). Unlike the kernel density estimator in which the number of terms is equal to the sample size, the adaptive mixtures estimator only adds as many terms as needed and that number of terms is potentially many orders of magnitude less than the sample size. Of course, as indicated above, I believe that fundamentally these very large data sets must be addressed nonparametrically since any parametric model will surely be shown not to fit for very large data sets.

DESIGNED SAMPLING An obvious suggestion for extremely massive data sets is a random subsampling. However, as I indicated above, this has the potential for obscuring interesting and perhaps extremely pertinent structure in the data. A simple

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November, 1994	Technical

4. TITLE AND SUBTITLE

Huge Data Sets and the Frontiers of Computational Feasibility

5. FUNDING NUMBERS

DAAL03-91-G-0039
N00014-92-J-1301
DAAH04-94-G-0367
N00014-93-1-0527

6. AUTHOR(S)

Edward J. Wegman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Center for Computational Statistics
George Mason University 4A7
Fairfax, VA 22030

8. PERFORMING ORGANIZATION REPORT NUMBER

Technical Report # 110

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy
Office of the Chief of Naval Research
Mathematical Sciences Division
800 N. Quincy Street Code 1111SP
Arlington, VA 22217-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Recently, Huber (1994) offered a taxonomy of data set sizes ranging from tiny (10^2 bytes) to huge (10^{10} bytes). This taxonomy is particularly appealing because it quantifies the meaning of tiny, small, medium, large and huge. Indeed, some investigators consider 300 small and 10,000 large while others consider 10,000 small. In Huber's taxonomy, most statistical and visualization techniques are computationally feasible with tiny data sets. However, with larger data sets computers run out of computational horsepower and graphics displays run out of resolution fairly quickly. In this paper, we discuss aspects of data set size and computational feasibility for general classes of algorithms in the context of CPU performance, memory size, hard disk capacity, screen resolution and massively parallel architectures We discuss some strategies such as recursive formulations which mitigate the impact of size. We also discuss the potential for scalable parallelization which will mitigate the effects of computational complexity.

14. SUBJECT TERMS

Computational complexity, interactive feasibility, HPCC, limits of visualization

15. NUMBER OF PAGES

25

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL